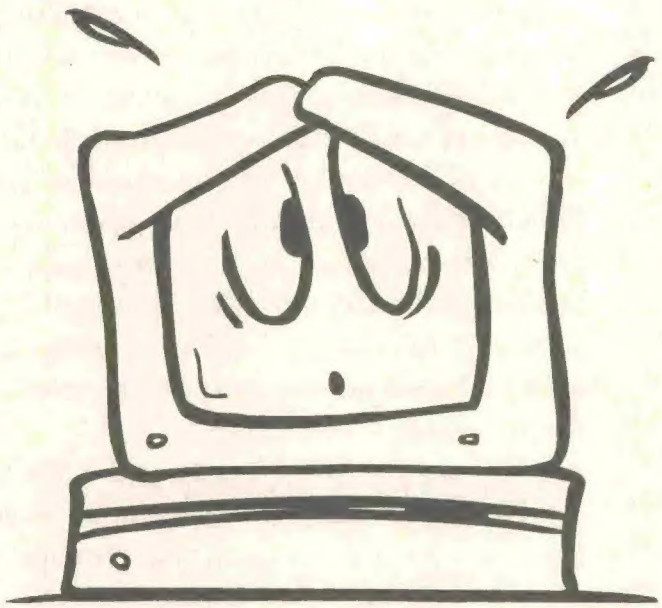
ธาวินทร์ ลีลาคุณารักษ์

Built-In event method ของ PARADOX



ถ้าเข้าใจแนวคิดต่าง ๆ ของการสร้างโปรแกรมแบบ VISUAL งานที่ได้จะมีประสิทธิภาพมากขึ้น

ทุกวันนี้ในวงการคอมพิวเตอร์มักจะมีเรื่องแปลกใหม่ ๆ ให้คอยติดตามกันอยู่เสมอ อย่างเรื่องของการสอนให้คอมพิวเตอร์จำเสียงของมนุษย์ได้ หรือถ้าจะพูดอีกอย่างหนึ่งคือ เราสามารถสั่งงานคอมพิวเตอร์โดยผ่านไมโครโฟนที่ติดตั้งกับเครื่องคอมพิวเตอร์ได้เหมือนกับการสั่งงานทางแป้นพิมพ์ ซึ่งการสั่งคอมพิวเตอร์ด้วยเสียงนี้เป็นเรื่องที่ได้รับความสนใจอย่างมาก โดยเฉพาะช่วงที่มีการเปิดตัวระบบปฏิบัติการ OS/2 Wrap เวอร์ชัน 4 เมื่อประมาณช่วงปลาย ๆ ของปีที่แล้ว ผมเองก็สนใจเรื่องของการสั่งงานคอมพิวเตอร์ด้วยเสียงนี้เหมือนกันครับ แต่ยังไม่มีโอกาสได้ทดลองใช้

อีกเรื่องหนึ่งที่ถือว่าเป็นเรื่องที่ค่อนข้างจะแหวกแนวจากความคิดความรู้สึกของคนคอมพิวเตอร์ทั่วไปก็คือ ในปัจจุบันนี้ คนที่จะทำการสร้างโปรแกรมคอมพิวเตอร์ไม่จำเป็นต้องมีความรู้ทางด้านโปรแกรมมิ่ง (Programming) เลยก็สามารถสร้างโปรแกรมคอมพิวเตอร์ที่มีประสิทธิภาพขึ้นมาใช้งานได้ ผมมีเพื่อนเป็นอาจารย์สอนในระดับมหาวิทยาลัยท่านหนึ่ง ซึ่งเขาได้ทำวิจัยเกี่ยวกับการจัดทำสื่อการเรียนการสอนบนคอมพิวเตอร์สำหรับเด็ก โดยใช้เครื่องมือเกี่ยวกับการจัดทำ Multimedia มาช่วยในการจัดองค์ประกอบและจัดลำดับของเรื่องราวต่าง ๆ ของงาน งานชิ้นหนึ่ง ๆ เขาจะใช้เวลาไม่นานนัก เมื่องานเสร็จสมบูรณ์จะได้เป็นโปรแกรมคอมพิวเตอร์สื่อการเรียนการสอนที่เป็น Executable file (.EXE) สามารถนำไปใช้บนเครื่องอื่น ๆ ได้ทันที โดยที่อาจารย์ท่านนี้ไม่มีความรู้เกี่ยวกับการเขียนโปรแกรมคอมพิวเตอร์เลย และงานที่จัดทำนี้ก็ไม่ต้องเขียนส่วนที่เป็นโปรแกรมเลยแม้แต่บรรทัดเดียว ลองนึกดูนะครับว่าถ้าเราใช้ภาษา C หรือ Pascal ในการจัดทำงานเหล่านี้เราจะเสียเวลาและใช้ความสามารถขนาดไหน

กลับมาดูที่งานเขียนโปรแกรมคอมพิวเตอร์ที่ต้องใช้ความรู้ทางด้านโปรแกรมมิ่งที่เป็นงานประจำของเราท่านบ้างนะครับ ทุกวันนี้หลาย ๆ แห่งเริ่มนำเครื่องมือหรือคอมพายล์เลอร์ประเภท Visual เข้ามาใช้กันบ้างแล้ว ผมเองก็พลอยได้รับความรู้ใหม่ ๆ นี้ไปด้วยเหมือนกัน

ปกติการสร้างโปรแกรมคอมพิวเตอร์ด้วยเครื่องมือประเภท Visual นี้ จะไม่เหมือนกับการสร้างโปรแกรมโดยการเขียนคำสั่งทีละบรรทัดจนกว่างานจะเสร็จอย่างที่เคยทำกันมาในยุคก่อน ๆ แต่การใช้เครื่องมือประเภทนี้เราจะใช้ทักษะทางด้านงานศิลปะมากขึ้น แต่งานด้านการเขียนชุดคำสั่งจะน้อยลง เพราะเราไม่ต้องเสียเวลาเขียนชุดคำสั่งในส่วนที่เป็นส่วน User Interface อีกต่อไป

โดยทั่วไปการสร้างโปรแกรมคอมพิวเตอร์ด้วยเครื่องมือหรือคอมพายล์เลอร์ประเภท Visual จะมีงานใหญ่ ๆ ที่ต้องทำอยู่ 2 งาน คือ งานสร้างส่วนประกอบของโปรแกรมที่ผู้ใช้สามารถมองเห็นด้วยตา หรือที่เรียกว่าส่วน User Interface ไม่ว่าจะเป็นส่วนที่แสดงบนจอภาพหรือส่วนที่เป็นรูปแบบรายงาน งานส่วนนี้จะเป็นงานที่ไม่ต้องเขียนโปรแกรมหรือชุดคำสั่งเลย แต่ต้องใช้ความสามารถทางศิลปะช่วยในการจัดทำ

สำหรับงานอีกส่วนหนึ่งคือการเขียนโปรแกรมหรือการสร้างชุดคำสั่งเพื่อควบคุมกลไกการทำงานของส่วนประกอบต่าง ๆ ของโปรแกรมให้ทำงานตามเงื่อนไขต่าง ๆ ที่เราต้องการ งานส่วนนี้ผู้ใช้จะมองไม่เห็น ซึ่งการสร้างชุดคำสั่งที่ผมพูดถึงนี้ เราจะทำเฉพาะบางจุดหรือบางเหตุการณ์ (event) ที่เราสนใจที่จะทำการควบคุมเท่านั้น ส่วนเหตุการณ์อื่น ๆ ที่เกิดขึ้นภายในโปรแกรมที่เราสร้างขึ้นที่เราไม่สนใจ เราจะไม่ไปยุ่งเกี่ยวกับเหตุการณ์ส่วนดังกล่าว แต่เราจะปล่อยให้เหตุการณ์ที่เราไม่สนใจนั้นดำเนินไปอย่างอิสระ ซึ่งการสร้างโปรแกรมในลักษณะนี้จะเรียกว่าการสร้างโปรแกรมแบบ event-driver นั้นเองครับ

Built-In event method ของ PARADOX

ผมได้นำเรื่องของ PARADOX มาพูดคุยกับท่านผู้อ่านมาหลายตอนแล้วนะครับ เหตุผลที่ผมนำเรื่องราวของ PARADOX มาเสนอเนื่องจากผมเองได้มีโอกาสใช้งานโปรแกรมชุดนี้ค่อนข้างบ่อย และ PARADOX ก็เป็นโปรแกรมประเภท Visual ตัวแรกที่ผมได้ใช้งาน

และความรู้ต่าง ๆ ที่ผมได้จากการใช้ PARADOX นี้เอง ทำให้ผมสามารถใช้โปรแกรมประเภท Visual ตัวอื่น ๆ ไม่ว่าจะเป็น Visual dBASE, Visaul Foxpro, DELPHI หรือ Power Builder ได้อย่างไม่ยากเย็น เพราะโปรแกรมเหล่านี้จะมีหลักการใช้งานคล้าย ๆ กันเพียงเวลาเราเรียกชื่อในรายละเอียดต่าง ๆ ในแต่ละโปรแกรมจะเรียกไม่เหมือนกันเท่านั้นครับ

ปัญหาอย่างหนึ่งของการสร้างโปรแกรมคอมพิวเตอร์ด้วยเครื่องมือหรือคอมพายล์เลอร์ประเภท Visual ก็คือ เราไม่รู้ว่าโปรแกรมที่เราสร้างขึ้นมานั้น เวลาที่โปรแกรมทำงานบนเครื่องคอมพิวเตอร์จะมีเหตุการณ์ (event) อะไรเกิดขึ้นกับส่วนต่าง ๆ ของโปรแกรมบ้าง

จากปัญหาตรงนี้เองทางผู้ผลิตคอมพายล์เลอร์เขาได้รวบรวมเหตุการณ์ต่าง ๆ ที่คาดว่าจะเกิดขึ้นมาจัดทำเป็นเหตุการณ์มาตรฐาน ซึ่งจะช่วยให้ผู้ที่ทำการสร้างโปรแกรมตัดสินใจที่จะเลือกเหตุการณ์ต่าง ๆ มาทำการเขียนชุดคำสั่งเพื่อควบคุมการทำงานส่วนต่าง ๆ ของโปรแกรมได้ง่ายขึ้น

สำหรับใน PARADOX จะเรียกส่วนเหตุการณ์มาตรฐานนี้ว่า Built-In event method โดยปกติเหตุการณ์ที่ถูกกำหนดให้เป็นเหตุการณ์มาตรฐานนี้จะมีกลไกการทำงานมาตรฐานของตนเอง (โดยทั่วไปผู้ผลิตจะสร้างเป็นชุดคำสั่งของภาษาที่ใช้เช่นใน PARADOX จะเป็น method ใน Visual dBASE จะเป็น Procedure เป็นต้น) ซึ่งถ้าไม่มีการแทรกชุดคำสั่งใด ๆ ลงไปใหม่ คอมพิวเตอร์ก็จะทำงานตามมาตรฐานที่ถูกกำหนดไว้สำหรับเหตุการณ์นั้น เช่น ถ้าเราไม่มีการแทรกชุดคำสั่งลงใน Object ที่เป็นปุ่มคำสั่งที่เราสร้างขึ้น (ใน DELPHI จะเรียกว่า componant) เวลาที่เราเรียกโปรแกรมขึ้นมาทำงาน (RUN) แล้ว ไปคลิ๊กที่ปุ่มคำสั่งดังกล่าว ดู Object ปุ่มคำสั่งนี้จะทำงานตามปกติหรือตามธรรมชาติของปุ่มคำสั่งคือมีการยุบตัวเหมือนกับปุ่มคำสั่งถูกกดลงไปจริง ๆ แต่จะไม่มีการทำงานใด ๆ ตามมา

แต่ถ้าเรากำหนดว่า เมื่อผู้ใช้คลิ๊กที่ปุ่มคำสั่งให้โปรแกรมแสดงไดอะล็อกบางอย่างออกมา (โดยการแทรกชุดคำสั่งลงไปควบคุมใน Built-In event method ที่ชื่อ pushButton ของปุ่มคำสั่ง) เมื่อเรา Run โปรแกรม และคลิ๊กที่ปุ่มคำสั่งเดียวกันนี้ ตัวภาพของปุ่มคำสั่งก็จะยุบตัวเหมือนกับถูกกดลงไปจริง ๆ และมีการแสดงไดอะล็อกในแบบที่เรากำหนดตามมาด้วย

สำหรับเหตุการณ์มาตรฐานต่าง ๆ ใน PARADOX จะมีการแบ่งประเภทของเหตุการณ์ต่าง ๆ (event) ที่เกิดกับ Object หนึ่ง ๆ (โดยทั่วไปจะเป็น Object ที่แสดงบนจอภาพ) ไว้ 3 กลุ่ม คือ

1. Internal Event เป็นเหตุการณ์ที่เกิดขึ้นภายในตัว Object ที่เราสนใจ ในตารางที่ 1 เป็นรายละเอียดของเหตุการณ์ต่าง ๆ ที่เกิดภายในตัว Object ขณะที่โปรแกรมทำงานบนคอมพิวเตอร์

ตารางที่ 1 Internal Event

Built-In Event	ลักษณะการเกิดเหตุการณ์ (เริ่มทำงาน)
open	จะทำงานเมื่อ Object (คอมโปแนนท์ในภาษาอื่น) เริ่มแสดงบนจอภาพครั้งแรก
close	จะทำงานเมื่อมีการปิด Object หรือเลิกแสดง Object นั้น ๆ บนจอภาพ
canArrive	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรม (tab stop) เลื่อนเข้าสู่ Object โดยจะทำการตรวจสอบว่าจะยอมให้เข้ามาทำงานภายใน Object นั้น ๆ หรือไม่
arrive	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรมเข้ามาอยู่ภายในตัว Object หลังจากที่ผ่านการตัวสอบของ canArrive แล้ว
setFocus	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรมเข้ามาใน Object หลังจากที่เกิดเหตุการณ์ arrive ไปแล้ว ตัว Object พร้อมที่จะดำเนินกิจกรรมอื่น ๆ ต่อไป (โดยเฉพาะ Object แบบฟิลด์ข้อมูล)
removeFocus	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรมกำลังจะมีการเคลื่อนออกจากตัว Object
canDepart	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรมกำลังจะเคลื่อนออกจาก Object โดยจะตรวจสอบว่าจะยอมให้เคอร์เซอร์เคลื่อนออกไปจาก Object นั้น ๆ หรือไม่
Depart	จะทำงานเมื่อเคอร์เซอร์ของโปรแกรมเคลื่อนออกจากตัว Object ไปแล้ว โดยจะต้องผ่านการทำงานของเหตุการณ์ canDepart ก่อน
mouseEnter	จะทำงานเมื่อลูกศรของเม้าส์เข้ามาอยู่ภายใน Object หรือลูกศรของเม้าส์เคลื่อนมาถึงตัว Object ที่เราสนใจอยู่
mouseExit	จะทำงานเมื่อมีการเคลื่อนลูกศรของเม้าส์ออกจากตัว Object (เคอร์เซอร์ของโปรแกรมไม่จำเป็นต้องเคลื่อนออกไป)
timer	จะทำงานเมื่อถึงช่วงเวลาที่กำหนดไว้ให้เหตุการณ์ timer ทำงานโดยเราจะกำหนดช่วงเวลาให้กับ Object ด้วยคำสั่ง (method) ที่ชื่อ setTimer ก่อน

ตารางที่ 2 External Event

Built-In Event	ลักษณะการเกิดเหตุการณ์ (เริ่มทำงาน)
action	จะทำงานเมื่อมีการเรียกใช้คำสั่ง action จาก Built-In event อื่น ๆ
error	จะทำงานเมื่อเกิดข้อผิดพลาดขึ้นในโปรแกรม (Object เพจ จะรับค่าของข้อผิดพลาดเป็น Object สุดท้ายก่อนแสดง massage)
init	จะทำงานก่อนที่จะเกิดเหตุการณ์ open (ใน PARADOX 7 ส่วนในเวอร์ชันก่อน ๆ จะเกิดภายใน open)
keyCha	จะทำงานเมื่อมีการเรียกจากเหตุการณ์ keyPhysical ซึ่งจะใช้งานเกี่ยวกับการตอบสนองการกดแป้นพิมพ์ที่ตรงกับเงื่อนไขของงานที่เรากำหนดไว้
keyPhysical	จะทำงานเมื่อผู้ใช้กดแป้นพิมพ์ของเครื่องคอมพิวเตอร์
menuAction	จะทำงานเมื่อผู้ใช้เลือกตัวเลือกตัวใดตัวหนึ่งจากเมนูของโปรแกรม
mouseClick	จะทำงานเมื่อผู้ใช้คลิ๊กที่ปุ่มซ้ายของเม้าส์
mouseDouble	จะทำงานเมื่อผู้ใช้ดับเบิลคลิ๊กที่ปุ่มซ้ายของเม้าส์
mouseDown	จะทำงานขณะที่ปุ่มซ้ายของเม้าส์กำลังถูกกด (ไม่ว่าจะเป็นการคลิ๊ก ดับเบิลคลิ๊ก หรือกดค้างไว้)
mouseUp	จะทำงานขณะที่ปุ่มซ้ายของเม้าส์ที่ผู้ใช้กดอยู่กำลังถูกปล่อยออก (ไม่ว่าจะเป็นการคลิ๊ก ดับเบิลคลิ๊ก หรือกดค้างไว้)
mouseMove	จะทำงานเมื่อมีการขยับตำแหน่งของลูกศรของเม้าส์ภายในขอบเขตของ Object ที่กำลังมีกิจกรรมอยู่ (ไม่เกี่ยวกับ Object ข้างเคียง)
mouseRightDouble	จะทำงานเมื่อมีการดับเบิลคลิ๊กที่ปุ่มขวาของเม้าส์
mouseRightDown	จะทำงานขณะที่ปุ่มขวาของเม้าส์กำลังจะถูกกดลงไป
mouseRightUp	จะทำงานขณะที่ปุ่มขวาของเม้าส์ที่ถูกกดอยู่กำลังจะถูกปล่อยออก
status	จะทำงานก่อนที่โปรแกรมจะนำ massage ต่าง ๆ ไปแสดงในแถบ status bar ของโปรแกรม

ตารางที่ 3 Spacial Event

Built-In Event	ลักษณะการเกิดเหตุการณ์ (เริ่มทำงาน)
pushButton	จะทำงานเมื่อผู้ใช้คลิ๊กที่ปุ่มคำสั่งบนจอภาพ (มีใน Object ปุ่มคำสั่งเท่านั้น)
newValue	จะทำงานเมื่อมีการแก้ไขหรือเปลี่ยนค่าของฟิลด์ข้อมูลที่อยู่บนจอหรืออาจจะเกิดจากการ Refresh จอภาพ newValue จะเป็นค่าที่อยู่ในหน่วยความจำ
changValue	จะทำงานเมื่อมีการแก้ไขข้อมูลภายในฟิลด์ข้อมูลที่ปรากฏบนจอภาพ แล้วกำลังจะเก็บค่าใหม่ลงไปแทนค่าเดิมของข้อมูล

2. External Event เป็นเหตุการณ์ที่เกิดขึ้นภายนอกตัว Object แต่มีผลกระทบกับตัว Object ที่เรากำลังสนใจโดยตรง เหตุการณ์ในกลุ่มนี้จะมีหลาย ๆ เหตุการณ์ที่เกิดจากการกระตุ้นจากผู้ใช้โปรแกรม ซึ่งรายละเอียดของ External Event จะแสดงในตารางที่ 2

3. Special Event เป็นเหตุการณ์ที่จะเกิดกับ Object บางประเภทเท่านั้น เช่นที่ปุ่มคำสั่งหรือฟิลด์ข้อมูลเป็นต้น สำหรับรายละเอียดของ Special Event จะอยู่ในตารางที่ 3

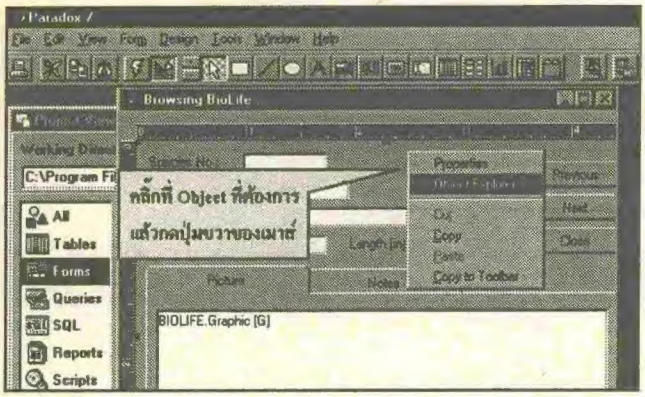
โดยทั่วไป Object ที่ปรากฏบนจอภาพที่ผู้ใช้โปรแกรมสามารถมองเห็นได้ มักจะมีทั้ง Internal และ External Event เกิดขึ้น จะมี Object บางชนิดเท่านั้นที่จะมี Special Event เกิดขึ้น และในแต่ละ Object หรือใน Object แต่ละแบบ จะมีเหตุการณ์ที่เกิดขึ้นกับตัวเองแตกต่างกันตามแนวคิด (concept) ประจำตัวของแต่ละ Object

ในรูปที่ 2 เป็นตัวอย่างของ Built-In event method ที่จะเกิดกับ Object ที่ชื่อ #NotePag5 ซึ่งเป็น Object ที่เป็นหน้าย่อย ๆ ของ Object ประเภท NoteBook ส่วนในรูปที่ 3 เป็นตัวอย่างการแทรกชุดคำสั่งลงใน Built-In event method ที่ชื่อ pushButton ซึ่งเป็นเหตุการณ์ที่เกิดขึ้นเมื่อผู้ใช้คลิ๊กที่ตัวปุ่มคำสั่ง เป็นเหตุการณ์ที่เกิดกับ Object ประเภทปุ่มคำสั่งเท่านั้น

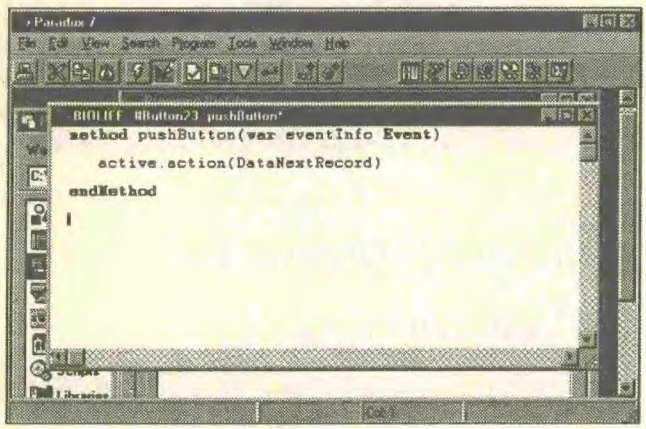
ลำดับของเหตุการณ์ที่เกิดขึ้นตอนโปรแกรมทำงาน

ปกติการจะพิจารณาว่าเหตุการณ์ใดเกิดขึ้นก่อนหลังหรือมีลำดับอย่างไร จะต้องใช้จินตนาการของผู้ที่ทำการสร้างโปรแกรมพอสมควร เพราะว่าในการทำงานของโปรแกรมมักจะมีองค์ประกอบอื่น ๆ เข้ามาเกี่ยวข้องด้วยเสมอ ตัวอย่างเช่น ถ้าเราพิจารณาเหตุการณ์ที่ชื่อ open เหตุการณ์นี้จะเกิดเมื่อ Object ถูกเปิดขึ้นมาแสดงบนจอภาพ

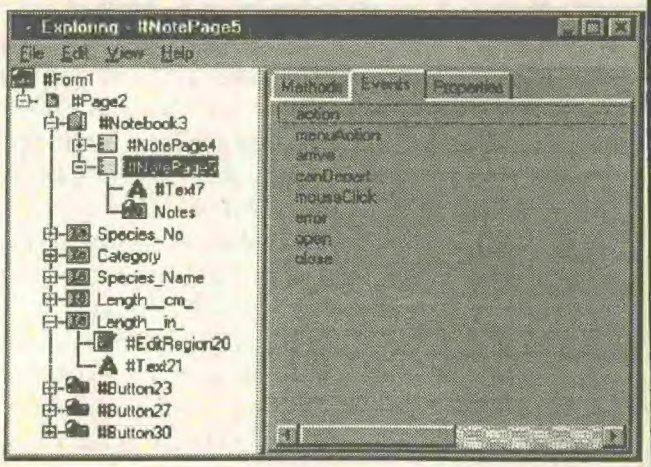
ถ้าเราดูตัวอย่างของ Object ต่าง ๆ ในรูปที่ 2 ประกอบจะเห็นว่าในทุก ๆ Object ที่อยู่บนฟอร์มที่เป็นหน้าจอของโปรแกรมประยุกต์ที่เราสร้างขึ้นชุดนี้ จะมี Built-In event method ที่ชื่อ open อยู่ทุก Object และเวลาที่เหตุการณ์ open ทำงานนั้น จะไม่ได้เกิดขึ้นพร้อมกันทีเดียวทุก Object แต่จะเริ่ม Object ที่ชื่อ #Form1 ก่อนแล้วไล่มาที่ #Page2, NoteBook3 ตามลำดับชั้นของ Object ที่ปรากฏบนโครงร่างแขนง (Object Tree) ของฟอร์มชุดดังกล่าว โดย Object ที่อยู่ระดับบน ๆ จะเกิดเหตุการณ์ open ขึ้นก่อน Object ที่อยู่ระดับล่าง ๆ ของโครงร่างแขนงจะเกิดเหตุการณ์ open หลังสุด



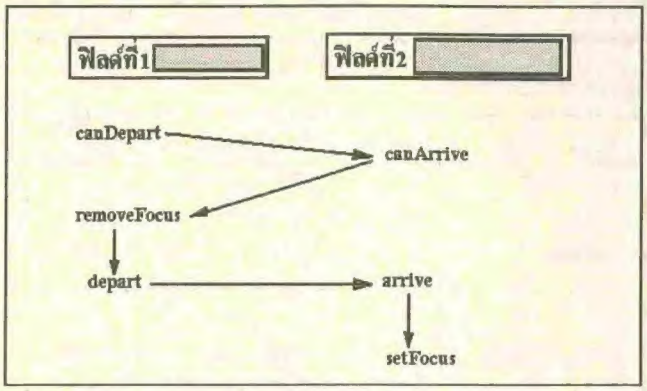
รูปที่ 1 ขั้นตอนการเรียก Object Explorer ของ Paradox 7



รูปที่ 3 ตัวอย่างการแทรกชุดคำสั่งลงใน Built-in event ของ Object



รูปที่ 2 ตัวอย่าง Object Explorer ที่แสดง Built-in event ของ Object



รูปที่ 4 ตัวอย่างเหตุการณ์ที่เกิดขึ้นขณะที่มีการย้ายเคอร์เซอร์จากฟิลด์หนึ่งไปยังอีกฟิลด์หนึ่ง

แต่ถ้าเราพิจารณาที่เหตุการณ์ที่ชื่อ arrive ซึ่งจะเกิดขึ้นเมื่อเคอร์เซอร์ (tab stop) ของโปรแกรมเข้ามาอยู่ภายในตัว Object พร้อมที่จะดำเนินการอื่น ๆ ต่อ เหตุการณ์นี้จะเกิดกับ Object ที่อยู่ในระดับล่าง ๆ ของโครงร่างแขนง (Object Tree) ของ Object ที่บรรจุอยู่ในฟอร์มเป็นส่วนใหญ่ โอกาสที่จะเกิดกับ Object ระดับบน ๆ หรือที่ root จะน้อยมาก ในรูปที่ 4 เป็นตัวอย่างของเหตุการณ์ต่าง ๆ ที่เกิดขึ้น เมื่อผู้ใช้กำลังย้ายเคอร์เซอร์ของโปรแกรมจากฟิลด์แสดงข้อมูลชื่อ ฟิลด์ที่ 1 ไปยังฟิลด์ข้อมูลที่ชื่อ ฟิลด์ที่ 2 ซึ่งเหตุการณ์เหล่านี้จะเกิดขึ้นต่อเนื่องกันจนถึงเหตุการณ์สุดท้ายของงาน

ในตัวอย่างที่ผมนำมานี้เป็นเพียงส่วนหนึ่งของเหตุการณ์ต่าง ๆ ที่เกิดขึ้นขณะที่โปรแกรมทำงาน แต่ในความเป็นจริงแล้ว ในฟอร์มชุดหนึ่ง ๆ จะประกอบด้วยหลาย ๆ Object ซึ่งถ้าจะดูลักษณะการเกิดเหตุการณ์ต่าง ๆ กันจริง ๆ แล้วค่อนข้างจะสับสนวุ่นวายมาก

แต่ในเวลาที่เราเขียนโปรแกรมหรือทำการแทรกชุดคำสั่ง เราจะยังไม่สนใจความซับซ้อนของการเกิดเหตุการณ์ต่าง ๆ แต่เราจะมองว่าเหตุการณ์ทุกอย่างที่จะเกิดขึ้นกับโปรแกรมจะมีระดับความสำคัญเท่ากันหมด และจะมองว่าเหตุการณ์ต่าง ๆ ที่จะเกิดขึ้นนี้จะเกิดตามลำดับเหมือนกับการที่เราเขียนโปรแกรมทีละคำสั่งจนกว่าจะจบงาน (การดูลำดับเหตุการณ์เราจะดูจากชื่อของเหตุการณ์มาตรฐานที่ผู้ผลิตคอมพายเลอร์กำหนดมาให้ ซึ่งจะบอกลำดับของเหตุการณ์ได้อย่างคร่าว ๆ)

และในการแทรกชุดคำสั่งลงในโปรแกรม เราจะค่อย ๆ ทำไปทีละ Object โดยเราจะเลือกเหตุการณ์ที่เราเห็นว่าควรจะเข้าไปควบคุม ส่วนที่ว่าเราจะเลือกหรือรู้ได้อย่างไรว่าเหตุการณ์ (event) ไหนเป็นเหตุการณ์ที่เราจะต้องแทรกชุดคำสั่งลงไปนั้น คงต้องศึกษารายละเอียดของแต่ละเหตุการณ์ (Built-In event method) ว่าเหตุการณ์นั้น ๆ เกิดขึ้นเมื่อใด แล้วพิจารณาว่าเราต้องการให้เกิดกิจกรรมอย่างไรกับ Object ที่เราสนใจอยู่บ้าง และเราจะกำหนดให้กิจกรรมนั้นเกิดขึ้นเมื่อไร เราจะได้เลือกควบคุมเหตุการณ์ต่าง ๆ ได้อย่างถูกต้องแรก ๆ คงต้องลองผิดลองถูกหันพอสมควรครับ

ทีแรกผมว่าจะพยายามให้ได้เนื้อหาที่เข้าใจง่าย ๆ หน่อย แต่พอถึงตอนนี้รู้สึกว่าจะค่อนข้างหนักไปแต่เชื่อว่าท่านผู้อ่านน่าจะได้ประโยชน์บ้างตามสมควรนะครับ ผมคิดว่าการสร้างโปรแกรมคอมพิวเตอร์ โดยใช้เครื่องมือประเภท Visual นั้น ค่อนข้างจะได้ผลงานออกมาเร็ว และได้งานที่มีประสิทธิภาพ ถ้าเราเข้าใจแนวคิดต่าง ๆ ของการสร้างโปรแกรมแบบ Visual เป็นอย่างดี